

The Official AMOS



CLUB NEWSLETTER Vol 2 Issue 3

Another AMOS Club Newsletter !
More new AMOS software and lots of listings and useful information for all you AMOS users.

TOME Series IV is released.
October 10th is the release date for AMOS TOME Series IV, the new version of the all powerful map system. This new version is available to existing users for £10 as an upgrade on Series 3, or separately for £24.99 (£29.99 for non AMOS Club members).

Upgrades for this one should be sent to Shadow Software/The AMOS Club. See details on Page 11.

AMOS Professional is released on 15th October 1992. If you already own AMOS, then it is available as an upgrade for £30 until the end of October, and £35 after that date. The upgrades are **ONLY available from Europress Software** (see the address in the last issue) not from the AMOS Club. As Pro wasn't finished at the time of going to press, we have only been able to have a brief look at the unfinished preview version. Details on Page 19.

The OZ Computer Happening '92

This is a computer show with a difference, happening in Belgium on November 7th & 8th. Instead of having wall to wall box shifters, selling discount software, the OZ

happening, which is sponsored by the Belgian Computer Press and the "Meetjeslandse Computer Club", is an event where visitors go along to meet and talk to the developers and creators of the software they use. Instead of shop stands, there will be "islands" with Amigas, and the programmers, who will be demonstrating new software, and explaining how to write, or use their latest masterpieces. As well as myself, representing the U.K./International AMOS Club (Demonstrating TOME Series IV, CText 2, Sprite X 2 and all the latest AMOS Club and Deja Vu toys, as well as some programming tricks), there will be representatives from the Dutch and German AMOS User

In This Issue:

The AMOS Helpline	3
Memory Magic	5
Professor Speck introduces probability (Probably !)	8
TOME Series IV Review	10
10 Liners (Time/Date, Pasta Roids and Roid Ball)	12
D-Sam Review	15
Extending AMOS	17
The Flashy Rem Generator	18
AMOS Professional Preview	19
AMOS Club products order form	20

groups, and Francois will probably be there too !

Commodore Belgium will be showing off the new Amiga 4000, Nico Francois (Who wrote Powerpacker), Sensible Software, Peter Stuer (Boot X and Viruskiller) and many other major league programmers will be there.

If you are able too, try and get there, as it looks like it will be an experience not to miss !

AMOS Optimiser

Hopefully available by Christmas is an Optimiser for AMOS by Raymond Price, which will take your compiled AMOS programs, and remove any unused code, saving around 20% in terms of memory and disk space, and of course, speeding up your programs even further. Obviously, there will be more details in the newsletter once this rather useful program is finished, and it looks like it will be available through the AMOS Club and Totally AMOS disk magazine.

CText 2.5

6 more commands have been added to the CText 2 extension, as well as another 2 pages to its manual and some extra demo programs. Anyone with CText 2.0 can send it in to be upgraded, simply send in the original disk, and 4 first class stamps, and we'll send you the new disk, and the extra pages for your manual. All new orders for Ctext 2 will receive Ctext 2.5, which includes Pixel Accurate word wrapping and the ability to change the colours in a font amongst its many other talents !

Also, AMOS Club members can now order Ctext 2 without having to send in a Ctext 1.32 for upgrading. If you

have Ctext 1.32 and bought it from the AMOS Club direct, then CText 2.5 will cost **£10** (mention you have Ctext 1.32) and if you don't have Ctext 1.32 or ordered it from elsewhere (and don't want to send it in) CText 2.5 is **£11**. See the order form on the back page.

Easy AMOS Upgrade

Unfortunately, the price given to us originally by Europress for the AMOS to Easy AMOS Upgrade was incorrect. The correct price is **£19.99** and not £15.

Extras and Disk Extension

Rod Pascoe has combined the commands of his two extensions into the Extras extension (#12), which will soon be available through the AMOS Club.

Another Useful Address for any of our German, Austrian or Swiss members. The Software Society, based in Ravensburg, are now distributors for AMOS TOME and the TOME Goodies Disks.

**The Software Society,
Schwarzachstr. 41,
7980 Ravensburg 19,
Germany**

The Disk Issue 3 will be released in the next week or so, and will contain both AMOS & Easy AMOS versions of the Rainbow Warrior utility (The bug fixed version), the 10-liner version of Tetris, and the latest version of the AMOS Club/ Shuffle extension with around 50 commands!

This Newsletter was written by and remains
© Aaron Fothergill except for stated articles
for which copyright remains with the original
author. No part of this publication may be
copied in any form, without prior consent by
Aaron Fothergill.

THE AMOS HELPLINE



The Helpline has been an integral part of AMOS Club membership, since AMOS was first released, and in the two years plus that it has been running, I have handled several thousand calls and solved a similar number of problems.

I thought it might be a good idea to give you some ideas of what the helpline has done over this time, how to get the best out of it, and what is going to happen to the helpline in the future.

One of the main advantages of the helpline is that it gives us a good idea of what people are doing with AMOS, what problems they are having (its amazing how everybody can have exactly the same problem on the same day !) and what they want adding to AMOS to make life easier. Ctext was written because of people requesting the system on the helpline, Sprite X evolved from it, the new version of TOME incorporates suggestions mentioned by users on the helpline and even the newsletter has changed according to what users are requesting (basically, *"give us as much information as possible, stop spelling useful as 'usefull' and make it readable !"*).

So your problem with AMOS might end up helping everyone, if you phone us up and ask how it can be solved. 99% of the bugs in AMOS are reported on the helpline too !

Because the helpline is actually run from my home it means that help is generally available around the clock if you are in desperate trouble (I run

my business from home, so there aren't any office hours to worry about). However, it isn't much use ringing when I'm out, asleep or watching a Grand Prix (or Indy car racing next year), so it's best to ring between 2pm and 7pm. This includes Saturdays and Sundays. There is less chance of me actually being available then, but if you need help, ring anyway ! Ringing after 7pm isn't a great idea, unless you are desperately stuck, and it can't wait until the following day.

Remember that ringing after 6pm is cheaper, although a lot of people will be doing this.

When you hit a problem with your programming, try and work it out for yourself first. Just like your teachers used to tell you, "You won't learn anything if you are just told how to do it !". Ringing me up at 10pm to ask for a program to make VU meters move to music will put you quite near the bottom of the popularity stakes !

Also, it will help me to help you if you can note down as much information about what is going on as possible. "It doesn't work" is not very helpful ! Note down where in the program the problem is occurring, what variables are being used, what values they are at when the routine fails (adding break points and printing them out helps). This way, I can get a good idea of what is happening in your program. This will save you (and me) a lot of time, and help keep your phone bill down !

Another thing that will save you a lot of time, and keep the phone usage down is to have a look through the newsletters to see if there is anything

related to what you want to know. As the newsletters contain a lot of programming requested by users on the helpline, it may be that someone else has had the same problem and it is sat there waiting to help you ! If it isn't, then after we've solved your programming problem it could go in a future newsletter to help everyone else.

When using the helpline, quote your membership number from the front of Volume 2 Issue 2, including the 2 digits printed in front. (e.g 5DX1490), as the people running the Europress helpline aren't particularly good programmers, they do tend to tell everyone to phone the Amos Club helpline, without mentioning that it is for members only. Thus we end up with a lot of non members (who usually want to become members, which is fair enough) blocking up the helpline, when valid, payed up club members need help. So to avoid any confusion, and me trying to get you to subscribe again, please quote your number when you phone !

If you get the engaged tone, try again 5 minutes later, it's usually someone else using the helpline, unlike radio phone-ins it isn't the first x callers that get help, so you don't have to sit there hitting the redial button like mad ! If it is a busy day, and the phone is engaged constantly, don't worry about phoning after 7pm, as long as it's a reasonable hour.

In the future, I'm going to try a few nights where I run the helpline to midnight, just for the heck of it. As a trial, on Wednesday the 28th of October, and Tuesday the 10th of November, you will be able to ring the helpline up to midnight. If this

system works, then it will be continued throughout the new year. I'll have just got back from the **Oz Happening** on the 10th of November, so if you are using the helpline that night you'll be able to find out what our overseas Amos users are doing, and what the new Amigas are like.

All in all, the helpline is there for all AMOS Club members, so make the best use of it. Try and solve any problems by yourself as much as possible, but:-

IF IN DOUBT, SHOUT !

you never know, even I might be having the same problem ! The AMOS Club Helpline can be contacted on **0271 23544**, preferably between 2pm and 7pm, never before 10am or after 10pm, unless it is required to save lives, the world or with news of a Mike Oldfield concert!

The AMOS Helpline

☎ 0271 23544

If you need help, phone !

Sprite X V2.0 is currently being worked on for release at Christmas. It has full screen editing of Bobs, Sprites, Icons, Screens and even Workbench Icons. Including an intelligent Bob Grabber (load in a screen and it works out where all the bobs are, grabs them and even puts hotspots on them !).

It is user friendly, and will be the most powerful image editor on the Amiga ! If you have any requests or ideas for bits you would like to see in it, phone them in on the Helpline !

Memory Magic

When you are writing your mega game in AMOS, you will often come across the messages "Out Of Memory", "Disk Full" or "Out of Variable Space". Before going into the details of methods of compacting the living daylights out of your data and programs, I'll explain how you can get these messages.

Out Of Memory

This will occur when you either run out of total memory (usually after fragmentation has occurred), or if you run out of Chip memory. When you run out of Chip memory, it is usually because you have too many screens open at once, too many graphics stored in memory (although Packed screens go into Fast ram) or too many samples and tunes floating around in memory.

Running out of total memory (fast and chip) can only be rectified by making the program, or its data smaller. If it occurred because of memory fragmentation, symptoms of which are out of memory errors when both Chip Free and Fast Free report plenty of memory, it is best to try and find the cause of the fragmentation (usually lots of screen opening and closing in quick succession, or strings being global'ed), and solve it before rebooting AMOS.

Disk Full

This can only be cured by losing files from the disk, or compacting them. If it is a disk that you are saving your AMOS programs onto, remember that AMOS will automatically save a backup of your program with .Bak instead of .AMOS on the end of the

filename. It is a good idea to either move the backups to a backup disk, or even to switch the Backups option off from the configuration editor (not such a great idea, as the auto Backup has saved me several times !).

Out of Variable Space

AMOS has a default variable buffer of 8K, if the variables and labels in your program use more than this, you will have to put a Set Buffer command at the start of your program to make it bigger. The only parameter for this is the number of K that you want to reserve for your variables. For instance, if you wanted to double it, you would put Set Buffer 16

It is best to either double the amount if you have added a large number of arrays or variables to a program, or to add 2K at a time when you are just adding bits to your program and the error occurs.

Compaction

Fortunately for AMOS users, there are many ways of compacting data. Screens, for instance, can be packed with the Pack and Spack commands, and when compiling you can pack the compiled program itself with the Compiler's inbuilt squasher. If you are using sprites, bobs or icons and can get away with the problems associated with it, you can use the image flipping routines built into AMOS to save on the number of images stored.

For instance, if you have several images where a flipped version is required, you can simply use =Hrev and =Vrev to display the flipped versions. Sometimes however, you might need to display flipped and unflipped at once, or don't want to

lose any speed. In which case, you can still save disk space by saving just one rotation of each to disk, and then reconstructing the bank by flipping and Get'ing these images upon loading.

If you are using TOME, you can pack the icon bank used for your tiles by pasting them back onto a screen (using the Tile_paster utility with TOME) and then Spacking it. Upon loading you simply unpack it and use the Get Icon command to get the tiles. e.g

```
Rem Load in a Packed screen
Load "MytilesPP.abk",5
Unpack 5 to 7 : Screen Hide 7
Rem cut them into tiles XSIZE
x YSIZE
XSIZE=16 : YSIZE=16
Erase 2
For B=0 to Screen Height/
YSIZE
For A=0 to Screen Width/XSIZE
Get Icon
Length(2)+1,A*XSIZE,B*YSIZE
to (A+1)*XSIZE,(B+1)*YSIZE
Next A
Next B
```

Samples unfortunately, cannot be easily packed without losing quality. They can be played from Fast Ram if you have D-Sam however. The best thing to do is to trim your samples very carefully when you create them, and to make sure that you only use as high a sample rate as you actually need.

The following routines can be used if you want to squash your bobs (it can be modified for icons also) or store them in Fast Ram.

The two procedures allow you to Crunch and Uncrunch a bob bank into any other bank.

Bob Crunching Routine,

By Aaron Fothergill

```
1) F$=Fsel$ (*.abk", "", "Load
a bob bank") [35]
2) Load F$ [68]
3) CRUNCHBOBS[5] [47]
4) Bsave "bobs",Start(5) To
Start(5)+V [53]
5) LDCRUNCHED["Bobs",5] [31]
6) UNCRUNCHBOBS[5] [53]
7) Procedure CRUNCHBOBS[BNK]
[48]
8) Rem BNK is the bank to
crunch into.. Returns V
(Length of crunched bank)
[66]
9) Shared V [119]
10) Dim SL(Length(1)) [83]
11) Rem Bob to Bank Convertor
[67]
12) ' [38]
13) ' [38]
14) ' Part 1, Calculate byte
length of bob bank [63]
15) ' [38]
16) L=Length(1) : Rem number
of bobs [42]
17) BL=0 [7]
18) For A=1 To L [109]
19) B=Sprite Base(A) [19]
20) W=Deek(B)*2 [62]
21) H=Deek(B+2) [32]
22) P=Deek(B+4) [62]
23) S=W*H*P+10 [0]
24) SL(A)=S [49]
25) BL=BL+S [69]
26) Next A [97]
27) ' [38]
28) ' Part 2, copy data for
each bob to contiguous
memory [63]
29) ' [38]
30) Erase BNK [38]
31) Reserve As Work
BNK,L*4+2+BL+64 [95]
```

```
32) SB=L*4+66 [84]
33) Doke Start(BNK),L [66]
34) For A=0 To 31 [47]
35) Doke Start(BNK)+L*4+2+
A*2,Deek( Start(1)+L*8+2+A
*2) [78]
36) Next A [97]
37) For A=1 To L [109]
38) Loke Start(BNK)+2+(A-1)
*4,SB [122]
39) Copy Sprite Base(A),
Sprite Base(A)+SL(A) To
Start(BNK)+SB [91]
40) SB=SB+SL(A) [73]
41) Next A [97]
42) ' [38]
43) ' [38]
44) V=Squash(Start(BNK),
Length(BNK),-1,512,17) [60]
45) End Proc [105]
46) Bsave "bobs",Start(5) To
Start(5)+Length(5) [75]
47) Procedure LDCRUNCHED
[B$,BNK] [114]
48) Rem B$=crunched bob file
to load, BNK=bank to load
into, returns V=length of
crunched file [41]
49) Shared V [119]
50) Erase BNK [38]
51) Open in 1,B$ [100]
52) BL=Lof(1) [71]
53) Close 1 [71]
54) Reserve As Work BNK,BL
[1]
55) Bload B$,Start(BNK) [40]
56) V=BL [97]
57) BL=Deek(Start(BNK)+BL-2)
[25]
58) Erase BNK [38]
59) Reserve As Work BNK,BL
[1]
60) Bload B$,Start(BNK) [40]
61) U=Unsquash(Start(BNK),V)
[27]
```

```
62) End Proc [105]
63) Procedure UNCRUNCHBOBS
[BNK] [42]
64) Rem BNK=bank to uncrunch
[38]
65) ' [38]
66) ' [38]
67) ' now recreate the sprite
bank [52]
68) ' [38]
69) ' [38]
70) ' [38]
71) L=Deek(Start(BNK)) [106]
72) SB=L*4+66 : HP=0 : BPL=0
[11]
73) For A=1 To L [109]
74) SB=Leek(Start(5)+(A-1)*4
+2) [46]
75) W=Deek(Start(BNK)+SB)
[89]
76) P=Deek(Start(BNK)+SB+4)
[64]
77) H=Deek(Start(BNK)+SB+2)
[94]
78) HP=Max(H,HP) [0]
79) Bset P,BPL [9]
80) Next A [97]
81) For BP=1 To 6 [73]
82) If Btst(P,BPL) [18]
83) SO=0 [21]
84) For A=1 To L [109]
85) SB=Leek(Start(BNK)+(A-
1)*4+2) [93]
86) W=Deek(Start(BNK)+SB)
[89]
87) H=Deek(Start(BNK)+SB+2)
[94]
88) P=Deek(Start(BNK)+SB+4)
[64]
89) BL=W*2*H*P+10 [79]
90) If P=BP [80]
91) If SO=0 [62]
92) SO=1 [20]
93) Screen Open 7,320,HP,2^
BP, Lowres [72]
```

```

94) End if [64]
95) Get Bob A,0,0 To W*16,H
   [93]
96) Copy Start(BNK)+SB,Start
   (BNK)+SB+BL To Sprite Base
   (A) [74]
97) Hot Spot A,Deek(Start
   (BNK)+SB+6),Deek(Start(BNK)
   +SB+8) [51]
98) End if [64]
99) Next A [97]
100) End if [64]
101) Next BP [53]
102) Screen Close 7 [98]

```

```

103) For C=0 To 31 [45]
104) Doke Start(1)+L*8+2
   +C*2,Deek(Start(BNK)+L*4
   +2+C*2) [78]
105) Next C [99]
106) End Proc [105]

```

These routines are obviously only really going to be useful to the power users out there, but they do allow you to save chip memory, as well as disk space. These routines are also included on issue 3 of "The Disk", along with 2 more procedures for uncrunching single bobs.

Professor Speck (Probably !)

He's Back ! The man who brought you psuedophysics and spaghetti coding returns with an article on probability, and how it will probably be useful in your programming ! **Most Basic programmers** will have used random numbers at some point, even if it is only to do a simple dice game. However, many programmers have not understood most of the basics of probability itself (as opposed to improbability, which it is claimed can make party hostesses underwear jump a foot to the left, and large white spaceships jump anywhere). As a student of the great probability theoretician Tonto Wun (who had the affliction of non-probability, which meant that any number he needed from his hyper accurate scientific recording die to prove his latest theory, would never occur), I have experimented with probability in computer games over the years, and found that random numbers are the basis of most great

games.

Basically, probability is working out the chances of something happening, either to make events happen at a certain level in a game, or to guess which bugs Francois will put in the next version of AMOS (amazingly enough, Francois manages to defy probability, and invent completely improbable bugs !). If, say, you were writing a space invaders type game, and wanted the each alien to have a small chance of releasing a bomb at any time, you would need to have some sort of routine that goes Eeny Meeny Miny Mo (the **EMMM**, as opposed to the **ERM** which is even more improbable than jumping large white spaceships across the galaxy) every frame to see if each invader should drop a bomb. In AMOS, the **RND()** function is incredibly good at this. You could simply do a line like:

```
If Rnd(99)=1 : Gosub DROPBOMB : Endif
```

which would basically mean that if the random number generation routine follows the rules of GCSE maths correctly, then it will come with a 1 once every 100 attempts on

average, so that the alien would drop a bomb roughly once every 4 seconds if the game is running at a normal 25 frames per second. However, as with most versions of Basic, the AMOS random number generator follows a slightly different law of probability, the **Law of Fat Chance**. This law states that a random number generator will tend to pick particular numbers more than others. This means that you shouldn't use it to run a computer roulette game, as someone is bound to spot the numbers that occur more often and make you incredibly poor (Unless you knew about this and put a cheat routine in). In the sort of routine above, this isn't as much of a problem.

By adjusting either the range of random numbers generated, or the amounts that will cause it to pass, you can change the probability of the routine triggering. For instance:

```
If Rnd(999999)=1 : Gosub DROPBOMB : Endif
```

would make the game very easy (except that according to Terry Pratchett, a 1 in a million chance will occur 9 times out of 10), and

```
If Rnd(1)=1 : Gosub DROPBOMB : Endif
```

will remind you to add the word **Aaargh** to your spellchecker ! The **EMMM** can be used in many situations, either as a direct method of picking what something will do, like in the Pac Dude game on the last issue of The Disk, or as a way of adding interesting little effects to a game. For instance, if your player is standing still, you can use an **EMMM** to make him blink, stamp his feet, scratch his head or wander off and

beat up the nearest thing that is moving (depending upon what sort of game you are writing). Little touches like this can really make a good game brilliant. Also, in adventure games, you can use an **EMMM** to annoy the living daylights out of the player by putting up messages like "You are sure you are being watched !", or "You hear a noise behind you....", which will make you very popular. When writing games or utilities based around rolling dice, either from converting an old board game or writing programs to help play these games, you have to watch out that you don't fall into the trap of thinking..

"I need to roll 3 dice and add them up. So I can do.. D=Rnd(18) giving a number from 1-18 !", as this falls foul of two factors. Number one is that using **Rnd(18)** will generate a number from 0 to 18, as AMOS recognises the mathematical fact that zero is the first number (a small splinter group of mathematicians is trying to prove that because -1 is less than zero, it should be regarded as the first number. They are however, racing to beat an American University team, who have discovered that the number -2 is even lower.). The other problem is that if you roll 3 dice, because of the way the numbers 1-6 are added together, you

a) *Get a range of 3-18 (The lowest you can get is 1+1+1)*

b) *The results tend to come out in the range 9-12 more often than the other numbers*

c) *you always get a 3 when you are rolling for Charisma, and have to walk along twenty yards ahead of the*

rest of the players in case you get ambushed.

So if you are doing dice, you should use the following for one 6 sided dice:

$D = \text{Rnd}(5) + 1$

and simply do this 3 times and add up the results for 3 dice:

$D = \text{Rnd}(5) + 1 + \text{Rnd}(5) + 1 + \text{Rnd}(5) + 1$

or $D = \text{Rnd}(5) + \text{Rnd}(5) + \text{rnd}(5) + 3$

If you are playing Dungeons and Dragons or any other role playing game the proper way (i.e. cheating!) you might prefer something along the lines of:

$D = \text{Rnd}(2) + \text{Rnd}(2) + \text{Rnd}(2) + 12$

Going back to the Law of Fat Chance, I did some tests on the level of probability of the `Rnd()` function returning a particular number in the range 1-100. As the routine progressed, calling the `Rnd()` function hundreds of thousands of times and noting down how often each number occurred, the program plotted up a graph of the occurrences of each number. After about 10 minutes, I looked at the result, and every time I ran the program, it looked suspiciously like a sound wave. I am now doing experiments to prove that if you generate numbers in a particular range, over a certain amount of time, feeding the results into the AMOS **Sam Raw** command will play the French Eurovision Song Contest entry.

Anyway, experiment with using random numbers in your programs, and remember the great sayings of Tonto Wun (who, according to his own theories probably didn't even exist!)

"I think therefore there's a one in five chance I might be" and "Go ahead dice, Make my day!"

TOME

Series 4

A Preview by Len Tucker.

Whether or not you already own TOME, this new release, TOME SERIES 4, is a real must. The added features that Aaron has applied make this as good as a brand new product of far superior power - and that's said by someone who has always thought that the old version of TOME was a dream come true!

Tome made the programming of mapped games really quick and easy, but now with commands like Map Handle taking over from the Nicemap subroutine, your programs will not only get better, but faster as well! Other commands such as Map FX, Map FY, Map HX and Map HY all help to make the user's life as easy as possible.

There are some stunning commands included, these are Map Anim, Map Fall and Map Swap Tile. Just these three commands make it worth upgrading to Series 4. If you have ever tried creating a static animation in Tome, you will have found out that it's not the easiest thing to achieve - e.g. how would you animate the flames on a torch in a dungeon game?

Map Anim makes things like this child's play. You use the TOME editor to say which tiles you want to animate, then you turn on an animation sequence in your program, and it's done automatically.

The wonders of TOME Series 4 must be seen to be believed, it's a lot more user friendly - I was well into using it just ten minutes after it arrived!

Map Fall - now this sounds a bit weird, but if you like Boulderdash or

Rockford type games, then this will allow you to write your own versions. It creates an avalanche of tiles as you see in these classic games. Of course it could be used to other purposes, for example to create a flood of water etc. Map Swap Tile - This command will allow you to flick between 2 tiles, e.g. door closed and door open.

Map Anim Move - now, not only can you animate tiles, but you can move them along the X and the Y axis.

List Tiles - This is a very handy little command which is used in Direct Mode. In this mode it will plot all the tiles in a bank onto the screen - this is a marvellous way of finding out where you put your tiles.

I'll leave you to find out for yourself just what TOME Series 4 can do. Think of working through TOME Series 4 as reading a good book, the last thing you'd want is to know every little detail - it's far better to explore it for yourself, this is just a taster to wet your appetite!

So far with TOME I have written an 8-way scrolling maze game (Marvin the Martian), a horizontal scroller (Flower Power) and my latest release, which is a commercial product and features an overview mapped driving game - all done with TOME. This goes to show that TOME can produce commercial quality products with ease.

As you can see, I've used TOME a lot, and think it's great, but I'm beginning to feel that I'm no longer needed (ah!). This is because there is so much that TOME can do automatically for you!

So, no matter what your ability - from expert to beginner -
YOU NEED TOME SERIES 4!!!!

TOME Series IV contains an **80 page printed manual**, covering every aspect of using the TOME system from a beginners tutorial, through to the full list of the **67 commands**, and full explanations of all the TOME Series IV Editor functions, including diagrams and screen shots. The TOME Extension, which adds 67 commands to AMOS, is automatically installed.

TOME Series IV also includes version 2.6 of the AMOS Club/ Shuffle extension, with a further 50+ commands, giving TOME Series IV nearly 120 new commands for AMOS!

The TOME Series IV editor is even more powerful (nothing comes near it!). It even has its own built in programming language for those power users out there! Other features include a Built in Animation Designer for Tile Animations, Improved Tile editing and Tile Maker facilities and an even higher level of user friendliness, including an improved niceness system (it will even remember the usual directory paths you use for each type of file). If you are a registered user of TOME Version 3 (3.0, 3.1 or 3.2) then **you can upgrade to TOME Series IV for only £10.00** (£12.00 overseas), just send in your original TOME disk, along with a cheque or postal order to **Shadow Software, 1 Lower Moor, Whiddon Valley, Barnstaple, N.Devon. EX32 8NW**. TOME Series IV costs **£24.99 to AMOS Club members** (£29.99 to non members) and is available directly from the AMOS Club / Shadow Software. See the Order form on the back page.

10 Liners

Typing in the following 29 lines of AMOS coding will leave you with

- (a) sore fingers
- (b) two games, and a date/time clock procedure !

Six Line Date/Time

By T.Powell

This nifty little procedure was sent in by T.Powell in Australia, and gives you full access to the Date and Time routines in the Amiga. The routine itself does the same job as Andrew Deeley's routine back in last year's issue 3 and Francois' Time/Date procedure provided with the updates, but it only uses 6 lines of code !

As usual, the listing is provided with Line numbers and Checksums, so that you can check it with the Checksum Checker. As usual, don't type in the line number and bracket, or the checksum in square brackets !

```
1) NOW [ 86]
2) Procedure NOW [ 58]
3) ML$="31283130313
0313130313031" : N$= Space$ (
12) : N=Varptr( N$) : Dreg(
12
```

```
1) =N [ 66]
4) C=Doscall(-192) : T=Leek(
N) : M=Leek( N+4) : TIC=Leek(
N+8) : YR=1977 [ 50]
5) While T>=DY : Add MTH,1,1
To 12 : L=1*(YR mod 4)=0 and
MTH=2 [ 43]
6) ML=Val(Mid$(ML$,MTH*2-
1,2))-L : Add YR,-1*(MTH=1) :
Add DY,ML [ 92]
7) Wend : DY=T+1-DY+ML : H=M/
60 : M=M mod 60 : S=TIC/50 [
46]
8) Print "Date: ";DY; MTH; YR,
"Time: ";H;M;S [ 76]
9) End Proc [ 105]
```

Obviously, you can modify the routine to pass the data back to your program, or just use it as it stands to print out the date and time.

Pasta roids

By Aaron Fothergill

Your ship was acting as an escort for a giant Cargo shuttle for the Acme Pizza Co. between Earth and Antares. Unfortunately, along the way, an anchovy and pepperoni flavouring meltdown occurred and the shuttle exploding, throwing huge chunks of pasta into space. You must now destroy all these Pastaroids before they hit any other ships!

O.k, so you've seen this one before ! This is the 10 line version of the classic Asteroids

game. Basically, you control your ship with left/right joystick, press the fire button to shoot and push the stick forwards and back to control thrust. As with the original game, your ship has full momentum (with a little damping). Have fun !

Pastaroids uses the graphics from the Amasteroids game on the AMOS Data disk, but if you want, you can create your own. Images 1-8 are the rotations for your ship, starting at 0° and rotating in 45° intervals. Images 17-24 are the explosion animation sequence, and images 25-32 are the animating asteroid.

```
1) Screen Open 1,320,204,16,
Lowres : Curs Off : Flash Off
: Cls 0 : Screen Hide : For
A=1 To 100 : Plot Rnd(319),
Rnd(199), Rnd(14)+1 : Next A
: Ink 1,0,1 : Screen 0 : Bob
Update Off [ 205]
2) Dim X(20),Y(20),R(20),
H(20),V(20),DX(7),DY(7) :
X=160 : Y=100 : R=0 : For A=0
To 7 : DX(A)=Sgn(Sin((A*Pi#)/
4.0)) : DY(A)=-Sgn(Cos((A*Pi#
)/4.0)) : Next A : L=4 : LV=3
: Autoback 0 [ 219]
3) Load "AMOS_DATA:Sprites/
Amasteroids.abk" : Curs Off :
Flash Off : Cls 0 : Get
Sprite Palette : Double
Buffer : Hide : BX=-99 : Get
Bob 57,0,0 To 1,1 : While D=0
```

```
: X=160 : Y=100 : DX=0 : DY=0
: SCO=-1 [ 70]
4) For A=1 To L : R(A)=Rnd(4)
: E=Rnd(1) : X(A)=E*Rnd(319)
: Y(A)=(1-E)*Rnd(199) : H(A)=
Rnd(8)-4 : V(A)=Rnd(8)-4 :
Next A : C=L : While C>0 and
D<8 : Bob 1,X,Y,1-(D=0)*R+D-
(D>0)*16 : For A=1 To L [
182]
5) Bob 2+A,X(A),Y(A),25+R(A)
: Bob 2,BX,BY,17+Rnd(1) : If
X(A)>-999 : Add R(A),1,0 To 7
: Add X(A),H(A),-8 To 328 :
Add Y(A),V(A),-8 To 208 : End
If : Next A : Screen Copy 1
To 0 : Bob Draw : Screen Swap
[ 44]
6) Add R,(Jleft(1)-Jright(1)
),0 To 7 : DX=DX-2*(Jup(1)-
Jdown(1))*DX(R)-Sgn(DX)*G/9 :
DY=DY-2*(Jup(1)-Jdown(1))*
DY(R)-Sgn(DY)*G/9 : Add X,DX,
0 To 319 : Add Y,DY,0 To 199
: Add G,1,0 To 9 [ 218]
7) If Fire(1) and BX=-99 :
BX=X : BY=Y : BDX=DX(R)*8 :
BDY=DY(R)*8 : BR=20 : End If
: If SC<>SCO : SCO=SC :
Screen 1 : Text 0,6, "P1:"+
Right$("000000"+Mid$( Str$(
SC),2),5) : End If : Screen 0
[ 78]
8) D=Max(D,-Bob Col(1,3 To
23)) : D=D+Sgn(D) : If BX>-99
: Add BX,BDX,0 To 320 : Add
BY,BDY,0 To 200 : B=Bob Col(
2,3 To 23) : If B : X(Col(-
2)-2)=-999 : Boom : Add SC,10
: BR=1 : Dec C : End If : Dec
BR [ 35]
9) If BR=0 : BX=-99 : End If
: End If : For A=0 To 9 :
M=(A mod 4)+9 : B=A*32 :
Screen Copy 1,B,8,B+32,M To
```

```

1,B,200 : Screen Copy 1,B,M,
B+32,204 To 1,B,8 : Next A :
Wend : LV=LV-Sgn(D) : L=L+1-
Sgn(D) [ 204]
10) D=(LV=0) : Wend : Screen
1 : Text 0,7,"GAME OVER" :
For A=0 To 71 : For B=1 To 7
: Paste Bob A*4,16+B*6,57-
Sgn(Point(A,B))*40 : Next B :
Screen Copy 1,0,16,320,80 To
Physic(0),0,32 : Next A [ 71]

```

I haven't tried compiling this one yet, as it's fast enough interpreted!

Roid Ball

By Aaron Fothergill

When the Pastaroids get bored they go out for a quick game of Roid Ball. Basically, this involves two players, each piloting a ship, and a Pastaroid fitted with a randomly controlled vector engine. The idea of the game is to tag the Pastaroid with your ship.

However...

The tag only counts if you are below a certain speed.

The Pastaroid has a tendency to change its direction when you least expect it.

Your ship has full momentum.

The other player can ram you, sending you whizzing off in the direction that he was going!

Of course, you can ram the other player too!

Control is exactly the same as in Pastaroids, except that you don't have the reverse thrust, and the game uses two joysticks (it being a two player game). First one to score 10 points wins!

```

1) Screen Open 1,320,204,16,
Lowres : Curs Off : Flash Off
: Cls 0 : Screen Hide : For
A=1 To 100 : Plot
Rnd(319),Rnd(199),Rnd(14)+1 :
Next A : Ink 1,0,1 : Screen 0
: Bob Update Off : RS=1 [
209]
2) Dim X(1),Y(1),R(1),VX(1),
VY(1), S(1),DX(7),DY(7) : For
A=0 To 7 : DX(A)=Sgn(Sin((A*
Pi#)/4.0)) : DY(A)=-Sgn(Cos(
(A*Pi#)/4.0)) : Next A :
Autoback 0 : Load "AMOS_DATA:
Sprites/Amosteroids.abk" [
172]
3) Curs Off : Flash Off : Cls
0 : Get Sprite Palette :
Double Buffer : Hide : While
S(0)<10 and S(1)<10 : BX=160
: BY=100 : BDX=0 : BDY=0 :
For A=0 To 1 : X(A)=16+A*256
: Y(A)=16+A*160 : Next A [
204]
4) Screen 1 : Paper 0 : While
RS=0 : Screen 0 : Add G,1,0
To 9 : Bob 1,X(0),Y(0),R(0)+1
: Bob 2,X(1),Y(1),R(1)+1 :
Bob 3,BX,BY,25+BR : Screen
Copy 1 To 0 : Bob Draw :
Screen Swap [ 171]
5) If Bob Col(1,2 To 2) : For
A=0 To 1 : X(A)=X(A)+VX(1-A)-
VX(A) : Y(A)=Y(A)+VY(1-A)-
VY(A) : Next A : For A=0 To 1

```

```

: VX(A)=VX(A)/1.5 : VY(A)=
VY(A)/1.5 : Next A : Swap
VX(0),VX(1) : Swap VY(0),
VY(1) : End If [ 177]
6) For A=0 To 1 : Add R(A),
(Jleft(A)-Jright(A)),0 To 7 :
Add X(A),VX(A),-8 To 310 :
Add Y(A),VY(A),-8 To 190 :
VX(A)=VX(A)-G/9*Sgn(VX(A)) :
VY(A)=VY(A)-G/9*Sgn(VY(A)) [
52]
7) VX(A)=VX(A)-Jup(A)*2* DX(
R(A)) : VY(A)=VY(A)-Jup(A)
*2*DY(R(A)) : If Bob Col(
A+1,3 To 3) and Abs(VX(A))
+Abs(VY(A))<16 : S(A)=S(A)+1
: Bell : RS=1 : End If [ 90]
8) Next A : Add BR,1,0 To 7 :
Add BX,BDX,-8 To 310 : Add
BY,BDY,-8 To 190 : BDX=Max(-
4,Min(4,BDX+Rnd(1)-Rnd(1))) :
BDY=Max(-4,Min(4,BDY+Rnd(1)-
Rnd(1))) [ 158]
9) For A=0 To 9 : M=(A mod
4)+9 : B=A*32 : Screen Copy
1,B,8,B+32,M To 1,B,200 :
Screen Copy 1,B,M,B+32,204 To
1,B,8 : Next A : Wend : RS=0
[ 148]
10) Screen 1 : Home : Print
"P1:";S(0) : Locate 30,0 :
Print "P2:";S(1) : Wend :
Screen 0 : Locate 0,8 : Paper
0 : Centre "Win By" : Paste
Bob 140,80,45-(S(1)=10) :
Screen Swap [ 205]

```

All these programs, along with "Tentris", the 10-line version of the classic Tetris game, and a complete game "Sabotage" are on issue 3 of "The Disk".
Have fun!

D-Sam AMOS sound

extension, by AZ Software:

Product review, by Aaron Fothergill
AMOS of course, has its own music and sound routines, built in.

However, there have been limitations with playing back samples, and few facilities for the tricks required when using larger samples.

The D-Sam extension adds 46 commands to AMOS, giving you almost total control over sample replay, including the ability to play samples direct from disk or hard disk.

The single disk, and 26 page manual are supplied in a plastic wallet, nice and sturdy, although the manual does tend to stick to the wallet a bit! The manual is a well presented A5 booklet, easy to read and understand. It covers basic aspects of sample replay before going in to detail on the commands, and this helps you understand what the commands are going to do.

I was quite impressed by the range of sample modes and types of sample that D-Sam supports. It can handle standard IFF samples, compressed samples, double speed samples, stereo samples, fades and software oversampling. It can also handle Audiomaster looping tables, enabling you to make up large pieces of sampled music from much smaller samples and it will also play samples from Fast memory as well as Chip memory, something which will make it popular with programmers who want all their chip memory for graphics!

D-Sam doesn't have to play its samples from disk, it can also work

from memory. Samples (and Audiomaser sequences) up to 400K long, and with replay rates up to an over 28KHz can be played from disk or memory (although the replay rate is much slower from floppy disk). The quality of playback is as good as any other Amiga playback routine I have heard, and the system is easy to use. Normally sample playback routines take up a lot of memory, but the D-Sam extension is about 16K in size, quite reasonable for the facilities it gives you.

D-Sam itself isn't used to do the sampling, for this you will need a good sampling package. Audiomaster II was given away free on a cover disk a couple of months ago. This combined with a reasonable sampling cartridge (around £20 for a cheap one) will give you a good enough system to use D-Sam. Most Amiga owners will already have some sort of a sampling package, especially if you are in to writing demos, for which I would highly recommend D-Sam ! D-Sam works well with AMOS, and as long as you have an up to date version of AMOS (1.34 with the fixed Music extension) you can even use the Music command in your programs, as well as the D-Sam commands (but not at the same time).

Although the manual covered the commands and use of D-Sam well, I would have liked to have seen more example programs on the disk. My review copy only had one example program and a compiled demo. The example did show off a lot of the commands, and most of the functions, but it would be better to have several examples, showing

different aspects of use. Maybe in a later version perhaps ?

Conclusion

I think that, as with most sound and music based software, most of the best results are going to come from the actual users of the software. Combined with a good sampling package, and a good knowledge of AMOS, D-Sam is going to be impressive. I think it is more likely to be used in Demos and multi media presentations, although AMOS written games will benefit greatly by using D-Sam. If you are working on anything that will require a good degree of sound quality, and or lots of samples, D-Sam is for you !

D-Sam is available from AZ software, for £19.99.

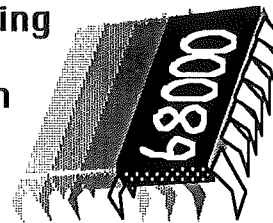
Order it direct from **AZ Software, AZ House, Broadfield Road, Eastington, Stonehouse, Gloucestershire. GL10 3BL**

Competition

AZ software, the producers of D-Sam have offered a copy of the software as a prize for this mini competition. To enter, answer the 3 questions below, and send them in, on a postcard, with your name and address to the AMOS Club, by November the 15th, 1992. The winner will be notified by post, and will be playing with D-Sam by Christmas !

- 1) Can D-Sam samples be played from Fast Ram ?
- 2) What is the highest pitch than can be heard by a normal human ear (A) 15Khz, (B) 44.1Kz (C) 20Khz
- 3) Which extension number does D-Sam use ? (for a clue, look in previous newsletters !)

Adding String & Number conversion to the Shuffle Extension.



By Aaron Fothergill

Most modern versions of basic have a set of functions to convert a number into a 1, 2 or 4 byte string, and then back again. When using integers in random access files this is almost vital. Unfortunately, Francois forgot them in AMOS (and it would seem in Pro to), so they have been added to the Shuffle extension.

Basically, load your previous source code version of the extension into your assembler (assuming you haven't got shuffle from "The Disk") and replace libraries 8-13 with the following code (these libraries were left blank in the last version).

The new commands are =MKB\$(n),=MKI\$(n) and =MKL\$(n) which turn byte, word and longword values into 1, 2 and 4 byte strings, and =CVB(n\$),=CVI(n\$) and =CVL(n\$) which turn them back into numbers again.

Add these lines to the end of the token list (before the dc.w 0 that ends the list).

```
@ dc.w @ -1,L_MKB
@ dc.b @ "mkb","$" + $80,"20"
,-1
@ dc.w @ -1,L_MKI
@ dc.b @ "mki","$" + $80,"20"
,-1
@ dc.w @ -1,L_MKL
@ dc.b @ "mkl","$" + $80,"20"
,-1
```

```
@ dc.w @ -1,L_CVB
@ dc.b @ "cv","b" + $80,"02",-1
@ dc.w @ -1,L_CVI
@ dc.b @ "cv","i" + $80,"02",-1
@ dc.w @ -1,L_CVL
@ dc.b @ "cv","l" + $80,"02",-1
*
```

Then add these lines in, replacing L8-L13 in the original L8

```
L_MKB @ equ @ 8
@ Dload @ a0
@ move.l @ (a3)+,d2
@ asl.l @ #8,d2
@ andi.l @ #$ff00,d2
@ move.l @ #1,d3
@ Rjsr @ L_Demande
@ move.l @ a0,-(a3)
@ move.w @ d3,(a0)+
@ move.w @ d2,(a0)+
@ move.l @ a0,HiChaine(a5)
@ move.l @ a1,d3
@ moveq @ #2,d2
@ rts
L9
L_MKI @ equ @ 9
@ Dload @ a0
@ move.l @ (a3)+,d2
@ andi.l @ #$FFFF,d2
@ move.l @ #2,d3
@ Rjsr @ L_Demande
@ move.l @ a0,-(a3)
@ move.w @ d3,(a0)+
@ move.w @ d2,(a0)+
@ move.l @ a2,HiChaine(a5)
@ move.l @ a1,d3
@ moveq @ #2,d2
@ rts
L10
L_MKL @ equ @ 10
@ Dload @ a0
@ move.l @ (a3)+,d2
@ move.l @ #4,d3
@ Rjsr @ L_Demande
@ move.l @ a0,-(a3)
```

```

@ move.w @ d3,(a0)+
@ move.l @ d2,(a0)+
@ move.l @ a0,HiChaine(a5)
@ move.l @ a1,d3
@ moveq @ #2,d2
@ rts
L11
L_CVB @ equ @ 11
@ move.l @ (a3)+,a0
@ clr.l @ d3
@ move.b @ 2(a0),d3
@ move.l @ #0,d2
@ rts
L12
L_CVI @ equ @ 12
@ move.l @ (a3)+,a0
@ clr.l @ d3

```

```

@ move.w @ 2(a0),d3
@ move.l @ #0,d2
@ rts
L13
L_CVL @ equ @ 13
@ move.l @ (a3)+,a0
@ move.l @ 2(a0),d3
@ move.l @ #0,d2
@ rts

```

Now assemble it again as Range.Lib and make sure that it is logged as extension #9 in AMOS. Issue 3 of "The Disk" contains the full shuffle V2.5 extension with around 50 commands, including commands to change bob and icon colours !

The Flashy Rem Generator

by Aaron Fothergill

The results of this program will impress, or annoy most people. Write yourself a program with the credits in rems at the start of the program, put *N* and *F* in the first two rem lines, then *F*, *N* and *** in the last three rems. Save it, run the following program and then load it again. The results will keep them guessing as to how you did it for ages !

```

1) ' Flashy Rem Generator[39]
2) ' *N* [72]
3) ' *F* [65]
4) ' Everything that is on
   these two lines will flash !
   [36]
5) ' This program is (C)
   Aaron Fothergill 1992 [81]
6) ' *F* [64]
7) ' *N* [72]
8) ' ***** [38]
9) F$="Flashrem.amos" : Rem

```

```

put your program title here
[81]
10) E$=Chr$(27): Open In 1,F$
   : L=Lof(1) : Close 1 [48]
11) Reserve As Work 5,L :
   Bload F$,Start(5) [66]
12) H=Start(5) : While
   H<Start(5)+Length(5)-4 [95]
13) H$="" : For A=0 To 2 :
   H$=H$+Chr$(Peek(H+A)) : Next
   A [64]
14) H2$="" [100]
15) If H$="*F*" : H2$=E$+"P3"
   : End If [50]
16) If H$="*N*" : H2$=E$+"P2"
   : End If [59]
17) If H$="***" :
   H=Start(5)+Length(5) : End
   If [58]
18) If H2$<>"" [120]
19) For A=0 To 2 : Poke H+A,
   Asc(Mid$(H2$,A+1,1)) : Next
   A : H=H+3 [88]
20) End If : Inc H : Wend[99]
21) Bsave F$,Start(5) To
   Start(5)+Length(5) [53]

```

AMOS Professional

Preview by Aaron Fothergill

AMOS Pro will be released on October 15th, and is the long awaited new version of the AMOS language. Unfortunately, it seems that Europress will not be giving out proper review copies until the release date (if you buy one on that date, the manual will probably still be warm from the printing press !), so I'm only able to do a preview of the package, based on the beta test version of the Pro Editor I was sent.

The AMOS Pro editor is very swish, it has the Workbench 2 flash buttons look, and the controls, such as load, run, test etc are now on a menu. This is actually better than it sounds, as it means you can use it to look up the keyboard shortcuts (I'm forever forgetting them), most of which are the same as in AMOS. The only disadvantage of this system is that you have to use a different system for marking blocks, although once learned, this is more of an advantage. The new direct mode is a major improvement, as the direct mode window can be resized with the mouse, and printing can be directed at it. Direct mode also uses the Shell system of recalling previous entries with the up/ down arrows. Pro also has a new monitor mode, similar to the one in Easy AMOS.

The Language itself is basically AMOS, with a few bits bolted on and a couple of minor changes. One of which is that the extension system is slightly different, and the extensions written for AMOS won't work with

Pro until re-assembled with the new equates table. Because this table is likely to change again when Francois writes the AMOS Pro Compiler in February, most of the third party developers (including the AMOS Club) won't be re-writing their extensions until then.

If you don't mind losing over 200 commands from the extensions, you get the capability of loading and playing Dpaint IFF Animation files, and a potentially powerful resource/ dialogue construction system, which unfortunately is based around strings, like AMAL. So they will be able to fall foul of the bugged string system, which Francois still hasn't fixed !

Although Francois has improved the speed of a couple of the commands, AMOS Pro is slightly slower than AMOS, although only by about 1-4% in most cases. According to the chaps at Europress, this is because the system itself is much bigger (AMOS Pro will only work on a 1 Meg machine).

The old AMOS Compiler, once you install the modified library for it, will actually compile AMOS Professional programs, as long as you don't use any of the new commands. Because of this, and the fact that you can't use the extensions, it would seem that it would be a good idea to wait until the Pro Compiler is released before deciding to buy AMOS Professional, giving Europress a good chance to fix the bugs and make it more efficient.

When released, AMOS Professional will cost £69.99 (£35 if you upgrade from AMOS)

Order Form

The number of products developed by the AMOS Club has got to the level where a whole newsletter could be produced consisting entirely of AMOS Club product reviews ! (Don't worry, If I do that, it will be a separate booklet, not an AMOS Newsletter !). By the next issue of the newsletter (hopefully out just before Christmas 92) there will be even more products, including the most powerful image editor ever, Sprite X 2, which will only be available to AMOS Club members.

Use this order form to send off for any of the AMOS Club products listed in the newsletter. Remember that as an AMOS Club member, you are getting a special discount price, and you are always able to get the latest version of any software written by the AMOS Club or Shadow Software.



Please Send Me....

- | | |
|---|--|
| ☐ AMOS TOME Series IV | £24.99 (Non Members £29.99) |
| ☐ TOME 3.2 to SIV upgrade | £10 +original disk (£12 Overseas) |
| ☐ AMOS CText V2.5 | £11 (£10 if Ctext 1.32 owned) |
| ☐ AMOS CText V1.32 | £2.50 |
| ☐ AMOS Sprite X 1.32 | £2.50 |
| ☐ "The Disk" Issue 3 | £2.00 (£2.50 overseas) |
| ☐ A Subscription to "The Disk" | £7.00 (£10.00 overseas) |

Name Membership #

Address

.....

.....

..... Postcode

Prices include VAT in the U.K. Prices also include Postage and Packing.

When ordering from overseas, please note that we can only accept cheques made out in pounds sterling, and drawn on a U.K bank (Eurocheques and American Post Office money orders made out in pounds can also be accepted). If ordering from Australia, Germany or the U.S.A, remember that all these products (except TOME Upgrades, Ctext 2.5 and The Disk in Germany and the U.S.A) are available from your local AMOS Club distributor, addresses are in issue 2, and this issue of the newsletter).

Please make all cheques payable to "The AMOS Club" and send your orders to **The AMOS Club, 1 Lower Moor, Whiddon Valley, Barnstaple, N.Devon. EX32 8NW.**